1.a. Give an example of a schedule showing the lost update anomaly.

=> When two transactions that access the same DB items and they have each of their operations interleaved leading to make some DB items incorrect

This is the order of every single operation occurred in each of two transactions

T1 : Read Item X

-> X = X + A

T2: Read Item X

-> X = X - B

T1 : Write item X

T2 : Write item X

The last queries of each transaction bring different answers even though they both require the same X value written as T1's update is lost while T2 is working on its own transaction.

1.b. Give an example schedule to show that the lost update anomaly is possible with the read committed isolation level.

Read committed isolation level does not allow dirty read – the transaction holds a read or write lock on the current row and prevents other transactions from reading, updating or deleting it

This is the order of every single operation occurred in each of two transactions

T1: (Read Item X with the Read committed isolation lock

T2: Read Item X

-> X = X – B

Commit

T1 : Read Item X

-> X = X + A

T1 : Write item X

T2 : Write item X) lock unleashed

T2's update is lost as T1 locked the other transactions until the lock released

2. You must explain how you reach to an answer. Writing a number without explanation is not accepted.

a) Calculate the record size R in bytes. (A record is composed of fields. A field holds information about an entity. For example, Name and SSN are two fields of an employee's record)

(30 + 9 + 9 + 40 + 10 + 8 + 1 + 4 + 4)+ 1 = 116

The amount of field bytes in total was calculated in the parenthesis, and the sum was added by an additional byte so 116 bytes

b) Calculate the blocking factor bfr and the number of file blocks b, assuming an un-spanned organization

* Blocking factor

The block size B in this disk => 512

The record size earlier => 116

512 / 116 = floor(result) => 4

*The number of file blocks

The file size = 30000

Its blocking factor = 4

After ceiling, the result is still 7500

c) Suppose that the file is ordered by the key field SSN and we want to construct a primary index on SSN. Calculate:

(i) the index blocking factor bfri (which is also the index fan-out fo);

Index record size = key field ssn (9) + block pointer(6) = 15

Index blocking factor = floor(block size(512) / index record size(15)) = 34

(ii) the number of first-level index entries and the number of first-level index blocks;

The number of first-level index entries=> 7500 (the number of blocks)

The number of first-level index block => ceiling(the number of blocks = 7500 / Index blocking factor = 34) => 221

(iii) the number of levels needed if we make it into a multilevel index;

*The number of second-level index entries = just the same amount as the first-level index block entries number => 221

And The number of second-level index blocks => ceiling(221/ index blocking factor = 34) = 7

*The number of third-level index entries = 7 (same as the second-level index blocks)

And ceiling(7 / 34) =...1

Therefore, three levels are needed if we make It into a multilevel

(iv) the total number of blocks required by the multilevel index; and

All blocks of each index block level = 221 first + 7 second + 1 third = 229

(v) The number of block accesses needed to search for and retrieve a record from the file—given its SSN value—using the one-level primary index and multilevel index.

It should go through 3 levels of indexes and third-level index entry 1 = 4

3. Consider the following relational schema. An employee can work in more than one department; the pct time field of the Works relation shows the percentage of time that a given employee works in a given department. [20 Marks]

Emp(eid: integer, ename: string, age: integer, salary: real)

Works(eid: integer, did: integer, pct time: integer)

Dept(did: integer, budget: real, managerid: integer)

Write SQL integrity constraints (domain, key, foreign key, or CHECK constraints) and SQL triggers to ensure each of the following requirements, considered independently.

1. Employees must make a minimum salary of $1000.

2. Every manager must be also be an employee.

3. The total percentage of all appointments for an employee must be under 100%.

4. A manager must always have a higher salary than any employee that he or she manages.

5. Whenever an employee is given a raise, the managers salary must be increased by the same percentage

```
CREATE TABLE Emp (
Eid INT PRIMARY KEY,
Ename VARCHAR(255),
Age INT,
Salary NUMERIC(9,2) CHECK ( Salary > 1000 )
);
CREATE TABLE Dept (
Did INT PRIMARY KEY,
Budget DECIMAL(10, 2),
Manager_Id INT REFERENCES Emp(Eid)
);
CREATE TABLE Works(
Eid INT REFERENCES Emp(Eid),
Did INT REFERENCES Dept(Did),
Pct_Time INT CHECK(Pct_Time < 100)
);
```